

A

jc660 U.S. PTO
04/26/99

THOMPSON HINE & FLORY LLP

Attorneys at Law

April 26, 1999

jc525 U.S. PTO
09/299502
04/26/99

In re Application of:

Inventor(s) : Forrest P. Gauthier
Filed : Herewith
Title : METHOD OF UTILIZING VARIABLE DATA FIELDS WITH A PAGE
DESCRIPTION LANGUAGE
Docket : 490334-002C2

BOX PATENT APPLICATION

Assistant Commissioner for Patents
Washington, D.C. 20231

U.S. EXPRESS MAIL AIRBILL NO.

EM425456724US

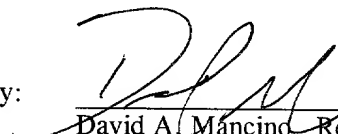
Sir:

Enclosed are the following papers:

- ☒ Specification, Claims and Abstract, 19 pages total.
- ☒ Drawings: 2 sheets. (Formal Informal).
- ☐ Declaration (Executed/Unexecuted).
- ☒ Priority Claim Under 35 U.S.C. §120 of U.S. App. Ser. No.08/896,899 filed July 18, 1997 and U.S. App. Ser. No. 08/373,582 filed January 18, 1995 and issued as U.S. Pat. No. 5,594,860.
- ☐ Statement by inventor or assignee as to reduced fees.
- ☐ Assignment and Recordation Form Cover Letter.
- ☐ Power of Attorney
- ☐ Other -

Respectfully submitted,

By:


David A. Mancino Reg. No. 39,289
(937) 443-6888

**METHOD OF UTILIZING VARIABLE DATA FIELDS
WITH A PAGE DESCRIPTION LANGUAGE**

Cross-Reference To Related Applications

This application is a continuation of U.S. Application, Serial No. 08/896,899, filed July 18, 1997, and issued as U.S. Patent No. _____; which is a continuation-in-part of U.S. Application Serial No. 08/373,582, filed January 18, 1995, and issued as U.S. Patent No. 5,594,860.

Background of the Invention

The present invention relates to the high-speed printing industry, and more particularly, to a method for printing variable data using a page description language in a high-speed printing environment.

Application programs, such as Adobe Illustrator®, typically include a program which generates a specification of a screen or page's contents in a page description language. The specification, or page description code, provides instructions as to how to generate the image in a printer. The page description code is transferred from the application program to a printer, where it is executed to generate a bit map of the page. The most commonly used page description language is PostScript®, which is a machine independent language produced by Adobe Systems, Inc.

An application program page typically contains a number of data areas with either graphic or alphanumeric data. The PostScript language includes commands that define or build "graphics states" for each of the data areas on the page. These graphics states are sets of default attributes such as angle, scale factor, type-font, location, etc., which define how data is to appear on the page. Often, multiple graphics states are defined for a single page, with the different graphic states corresponding to different data areas on the page. Examples of commands that are used in PostScript to build a graphics state are: 20 rotate, /Times-Roman findfont, 14 scalefont, and setfont. In addition to commands which build graphics states, PostScript specifications also include the

Docket No. 490334.002.C2

graphic or alphanumeric data which is displayed in the data areas, as well as a print command such as "SHOW", which causes a bit map to be generated for the data.

5 In the past, page description languages, including PostScript, have only been used to print static data pages, because page description languages lack the functionality required for variable data printing. In variable data printing, each page shares a common background, and the displayed data in at least one data field changes for each page. Up until now, it has not been possible to print pages of variable data with page description languages such as PostScript, because the page description languages are unable to save page backgrounds and graphics states from a page specification, and are thus unable reuse the same background and graphics states when printing subsequent pages. Thus, with page description languages such as PostScript, whether the entire page is changed, or only a single item of data on the page is changed, a new page description language specification is generated to print each separate page.

10 For example, if thousands of copies of a mass mailing advertisement were to be printed, each copy being identical except for the recipient's name and address, it would be necessary to generate a new PostScript specification defining the page background, and the graphics states for the name and address fields, for each new name and address that is printed. Hence, to print 50 advertisements, it would be necessary to generate 50 PostScript specifications which each define virtually the same image.

15 In general, PostScript specifications are very complex and require extensive processing to generate and execute. Thus, generating a new PostScript specification each time a page of variable data is printed consumes an immense amount of processing time. In high-speed printing systems, it is typically the processing time, not the printer hardware, which determines the speed at which pages can be printed. Therefore, the processing required to repetitively redefine the same background and graphics states for each page of variable data significantly slows the entire printing system.

20 Due to the amount of processing time consumed in redefining the page template and graphics states for each new page of data that is printed, as well as the resultant effect on printing

speed, it is desirable to have a method for processing variable data wherein once defined, the template and graphics states for a page can be stored and reused for printing subsequent pages. Further, it is desirable to have a method for printing variable data which is compatible with existing printing systems and page description languages, such as PostScript, and which is capable of processing variable data in a high-speed industrial printing system.

Summary of the Invention

It is an object of the present invention to provide a method for utilizing variable data with a page description language, which enables the template and graphics states for a page of variable data to be defined and stored; and which enables the stored graphics states to be associated with multiple items of variable data from a database or merge file; so that once stored, the graphics states can be repeatedly applied to the items of variable data to print multiple pages of variable data or multiple variable data bitmaps. Further, it is an object of the present invention to provide such a method which is compatible with existing page description languages, and which can be used in a high-speed industrial printing system.

The method of the present invention is implemented by means of a control task which executes in conjunction with a page description code interpretive program, such as a PostScript program, to identify variable data areas in the page description code specification, and reserve the graphics states for the variable data areas as they are defined by the specification. After the interpreter program has executed, a merge task is initiated. The merge task associates items of variable data from a data file with the reserved graphics states, generates a bit map for each variable data area, merges the bit maps with the page template, and outputs a complete bit map for the page. Accordingly, in the method of the present invention, bit maps for multiple pages of variable data are generated from a single page description language specification.

The present invention assumes the generation of a page specification in PostScript, or another similar page description language, by the application program, and the transfer of this

Docket No. 490334.002.C2

specification to a printer. According to the present invention, a control task activates and monitors the PostScript interpreter program in the printer. As the interpreter executes, it defines graphics states for the data areas on the page. The PostScript attributes for a graphics state are stored in a stack as they are defined, so that at any given point in the code, the stack represents all of the PostScript attributes for the current graphics state.

When the control task identifies a print command in the code, the control task interrupts the interpreter to determine whether the data to be printed is variable data. If the data is variable, the current graphics state, consisting of the attributes then existing in the stack and job specific attributes which are defined in a job file, is linked to the data area and reserved in an internal database. Further, character bit maps are generated in accordance with the graphics state, and linked to and reserved with the graphics state. After the graphics state and character bit maps have been reserved, the PostScript interpreter is resumed at the line of code following the print command.

The interpreter continues executing until either the control task detects another print command, or the last line of code is reached. If a second print command is detected, the interpreter is interrupted again and the above steps repeated, to reserve the stack contents and job attributes for the second data area, and to generate and store a second set of character bit maps. The control task continues in this manner monitoring and interrupting the interpreter program, until all of the variable data areas on the page have been detected, and graphics states and possibly character bit maps for the variable data areas have been reserved in the database.

As the PostScript interpreter executes, a bit map of the non-variable background graphics and text, otherwise referred to as a "template", is generated for the page. At the last code command, which in PostScript is typically "SHOWPAGE," the control task terminates the PostScript interpreter, and reserves the template in the database.

The merge task is then initiated to print variable data pages using the reserved page template, graphics states and character bit maps. The merge task begins by retrieving a merge file containing the variable data to be printed. After retrieving the merge file, the task identifies the correct template for the current page, and the names of the graphics states related to that template,

from data in the merge file. Then, using the name of the first graphics state reserved for the template, the merge task retrieves the graphics state from the database and the character bit maps linked to that state. The merge task then retrieves data corresponding to that graphics state from the appropriate field in the merge file, and generates a bit map of the data in accordance with the graphics state and character bit maps. The merge task then merges the data bit map into the template. After the bit map has been generated and merged, the merge task identifies retrieves another graphics state for the template and repeats the process. If there are no more graphics states which correspond to variable data areas on the page, the merge task outputs the finished bit map for the page.

After the first page of data has been printed, the merge task retrieves a "clean" template from the database, and again identifies the graphics states for the page. The merge task then retrieves the next record of variable data from the database, and generates variable data bit maps for each of the fields in the record, in accordance with the reserved graphics states and character bit maps which correspond to each of the fields. The merge task continues in this manner, identifying variable data areas and generating bit maps for the variable data in the merge file, until a page has been printed for each variable data record in the file.

The method of the present invention is advantageous in that once the graphics states and template have been defined for a variable data page, they can be reused to print multiple pages of variable data with only a minimal amount of additional processing.

Accordingly, it is an object of the present invention to provide a method for printing variable data with a page description language; a method which increases the speed at which variable data pages can be printed; a method which enables the printing attributes for a page to be saved and used for printing multiple pages of data; and a method which is compatible with existing page description languages and printing systems.

Other objects and advantages of the present invention will be apparent from the following description, the accompanying drawings and the appended claims.

Brief Description of the Drawings

Fig. 1 is a data flow diagram illustrating the preferred embodiment of the method of the present invention;

Fig. 2 is an example of a variable data page generated in accordance with the method of the present invention.

Detailed Description

The present invention provides a computer-implemented method for reserving graphics states, which enables a page description language to be used for variable data printing. In accordance with the present invention, graphics states describing variable data areas are generated by a page interpreter program and reserved in an internal database. The graphics states are later accessed from the database and used for printing variable data pages. The method of the present invention can be employed with a conventional page description language, such as PostScript, to enable variable data pages to be printed with a minimum amount of processing.

As shown in Fig. 1, an image containing text and/or graphics data is created at a workstation 10, using a graphics application program such as Adobe Illustrator®. As the image is created, the application program displays the image on the workstation screen. When the image is complete and ready to be printed as a page, the application program generates a specification of the image in PostScript in a conventional manner.

After the PostScript file 11 is generated, it is transferred from the workstation 10 to a printer generally designated as 12. In the printer 12, a PostScript interpreter 14 is executed to generate a pagemap of the image. In the method of the present invention, a control task operates in the printer 12 to initiate the PostScript interpreter program 14 and a merge task 16. The control task is initiated upon power-on of the printer 12, and controls and coordinates the variable data printing.

Docket No. 490334.002.C2

As the PostScript interpreter 14 executes, it defines the PostScript graphics state attributes for the page. These attributes can include the size, font, position, orientation, and location in which the graphic or text data is to appear on the page. The specifics of the attributes which are available in PostScript to define how data is to appear on a page would be well-known to one skilled in the art. Therefore, further description of these PostScript attributes is not provided.

However, one of the PostScript attributes, namely the region, has been expanded in the present invention to allow for variable data printing. In the method of the present invention, the region attribute is used to define the boundaries or extent to which a variable data graphics state will be allowed to effect a page. The graphics state extent is an invisible boundary which acts as a clippath for the page, to prevent variable data for a particular graphics state from extending outside the intended boundaries of the graphics state. The region of the graphics state extent is defined without altering PostScript, by using an ordinary shape, which is created as part of the image, to define the region. In the present invention, the artist creates a shape representing the extent as part of the page image, and specifies a particular trigger color or gray scale for the shape. Thus, the artist could include a medium gray scale rectangle in the upper left-hand corner of the page, with the boundaries of the rectangle representing the extent which the artist has defined for the graphics state positioned at that corner of the page. The medium gray scale will then represent the trigger color, and will be specified as the trigger for the region attribute in a job file 18 in the printer 12.

In addition, a second parameter in the job file 18 can be used to specify whether the rectangle should appear on the page, or whether it is being used only to define a graphics state extent. Thus, if the artist also wants the medium gray scale rectangle to appear on the printed page, this parameter enables the color to act as a trigger, yet not inhibit the artist's design. When the rectangle is interpreted during the method of this invention, the control task will detect the trigger color and will save an "invisible" boundary represented by the rectangular as part of the graphics state.

As the PostScript attributes are defined, they are placed in a stack. When a new attribute is defined, it is added to the top of the stack. When an attribute is deleted, it is removed

Docket No. 490334.002.C2

from the stack. The combination of all of the attributes located in the stack at any point during the execution of the PostScript interpreter 14 constitutes the "current" graphics state for the page.

When the interpreter reaches a print command, such as "SHOW" in PostScript, the command triggers the control task to interrupt the interpreter program. During this interruption, the control task interprets data in the PostScript file 11 and reserves a graphics state if the data is variable. Normally in a PostScript file, data which is to appear on the printed document is enclosed within parentheses. Thus, the control task identifies data in the file 11 by locating parentheses in the code.

After the control task identifies the data, it interprets the data to determine whether it is static data, which is to be part of the page template, or variable data. To interpret the data, the control task first reads the data located in the parentheses and compares the data with a list of literal data strings stored in the job file 18. The job file 18 contains a list of data strings which are each associated with the name of a graphics state and its corresponding data field in a merge file 20. In the preferred embodiment, the graphics state name is the same as the field name in the merge file 20. The merge file 20 contains variable data arranged in records, with each record corresponding to a different page. Each record contains one or more data fields, which each correspond to separate variable data areas on the page. The list of data strings and associated graphics state names is entered in the job file 18 by the print operator prior to initiating the print job. If the data from the PostScript file 11 matches a data string in the job file 18, the control task replaces the data from the file 11 with the graphics state name associated with the matching data string. In this manner, the control task transforms static data in the PostScript file into a variable data field, by substituting a graphics state field name for the static data in the file.

In a second embodiment, the graphics state name corresponding to the data area is defined directly within the PostScript file 11, by making the name part of the image that is created in the application program. In this embodiment, the name is enclosed within brackets in the file, such as "<< >>", to enable the control task to identify the data as defining a graphics state rather than being an ordinary data string. Thus, to define the graphics state "ADDRESS" within the PostScript

file 11, the following would appear before a show command in the code: "<<ADDRESS>>". This second embodiment is advantageous in that it does not require the control task to compare the file data with a data list in the job file 18; however, it does require coordinating the graphics state and field names between the merge file 20 and the application program.

5 If the control task determines that the data corresponds to a variable area, it reads the current contents of the graphics state stack to determine the attributes to be used for printing data in that area. In addition to the PostScript attributes specified in the stack, the graphics state can also include attributes which are specifically tailored to variable data printing. These additional attributes can either appear after the graphics state name inside a "show" command, if the graphics state is defined directly in the PostScript file, or can be specified in the job file 18 prior to execution of the print job. These additional attributes specify how the variable data is to be positioned within the graphics state. The following is a list of the variable data attributes which can be specified for a print job:

10 Name:

15 A label used to identify the data to which the graphics state applies. A single datum may be inserted into more than one graphics state so this attribute is not unique to a single state.

 Glyphs:

20 A list of character glyphs, both attributes and images, which are available for use in the graphics state. (e.g. an alphabet of 72 point Times-Roman bold italic characters).

 Static Data:

 Data to be used in the event that variable data is not available.

 Identification:

25 A number used to uniquely identify a graphics state.

 Justification:

Docket No. 490334.002.C2

How to handle the text left to right - left border, right border, centered or justified.

Alignment:

How to place the text vertically in the graphics state. This could be top, bottom or centered.

Word Wrapping:

Selects a word wrapping algorithm.

Dynamic Registration:

Information on how to determine the registration from one page to the next.

Logic Mode:

The manner in which the bitmap merge takes place. This is one of seven binary combination techniques.

DP Procedure:

A procedure (or program) used to manipulate the variable data just before the graphics state is applied.

Data Selection:

Which portions of the variable data to use.

Underline:

Selects underlined text.

When the control task is triggered to reserve a graphics state, the above listed attributes, if specified, are combined with the PostScript attributes from the stack, and reserved as a single graphics state under the name obtained from the PostScript file 11 or the job file 18 such as shown at 22.

After the control task has compiled the attributes for the current graphics state, it may instruct PostScript to generate a font cache 26 for the graphics state. The font cache 26 consists of a character bit map for each of the alphanumeric characters A-Z and 0-9 generated in the font

specified in the graphics state. After PostScript has generated all of the character bit maps, and placed the bit maps in the font cache 26, the font cache is linked to the graphics state 22, and reserved in the database. After the control task has reserved the current graphics state 22 and the font cache 26 in the database, it resumes execution of the PostScript interpreter 14 at the first line of code after the print or "SHOW" command, so that the print command is not executed.

After the interpreter is resumed, it continues defining graphics state attributes for the page, until the control task detects another print or "SHOW" command. Upon detecting another print command, the control task again interrupts execution of the interpreter, and determines whether the data in the PostScript file 11 corresponds to a variable data area. If the data corresponds to a variable data area, the control task again substitutes a graphics state name from the job file 18 for the data in the PostScript file 11, and reads the graphics state attributes from the stack and job file. The control task also instructs PostScript to generate another font cache, if the attributes of the current graphics state differ from the attributes of previously reserved graphics states. The current graphics state and font cache are then linked, and reserved in the database under the second graphics state name from the job file 18, such as shown at 24. If the data does not correspond to a variable data area, the control task resumes execution of the interpreter at the print command, so that a bit map for the data can be generated and added to the template.

At the final line of code, the template is complete, and incorporates all of the static text and graphic data that is to appear on the printed document. At this point, the control task terminates the interpreter, and saves the template to the database such as shown at 28. In PostScript, the control task is triggered to save the template by the "SHOWPAGE" command.

Since the control task of the invention operates externally of the PostScript interpreter, the method of the present invention enables bit maps and graphics states to be generated by the interpreter in a conventional manner. However, rather than printing a completed page map at the end of the interpreter program, the method of this invention reserves the page maps, character bit maps and graphics states generated by the interpreter, in order that they may be subsequently accessed and used to print multiple pages of variable data.

After the interpreter has been terminated, the control task initiates the merge task 16. The merge task 16 interfaces between the merge file 20, which has been pre-programmed with items of variable data, and the database in which the templates, font caches and graphics states defined by the interpreter have been saved, in order to combine the variable data with a template on a single page map. The merge task 16 begins by accessing the merge file 20 to retrieve the name of the template for the page, and then retrieving the specified template from the database. In addition, the merge task 16 retrieves the names of the data fields and reserved graphics states which are associated with the selected template from the merge file 20.

Using the name corresponding to the first graphics state on the page, the merge task 16 accesses the merge file 20 and retrieves the data stored under that field name in the first data record. In the representative merge file 20 shown in Fig. 1, the field names are NAME and NUMBER.

After the merge task 16 has read the data corresponding to the designated field name, it retrieves the graphics state which was reserved under the same name, as well as the character bit maps which are linked to that graphics state. The merge task 16 then generates a bit map of the data in accordance with the graphics state attributes. After the bit map is generated, it is merged into the template at the region corresponding to the graphics state, by writing the data bit map over the existing template bit map.

It will be apparent to those of ordinary skill in the art that it is within the scope of the invention to write the data bit map over a clean page as opposed to the template bitmap. For example, if the template contains no static bitmap data, then it would not be necessary to save an empty bitmap of the template in the database as described above. Thus, it is within the scope of the invention that the PostScript file 11 defines only variable data areas and does not define any static data areas. Such a PostScript file is illustrated in Fig. 1.

After the data from the first field has been merged into the template, the merge task 16 reads the name corresponding to a second variable data area from the merge file 20, if a second variable area exists on the page. The merge task 16 then retrieves the graphics state and linked font

Docket No. 490334.002.C2

cache having the same name as the second variable area. Next, using this name, the merge task 16 again accesses the merge file 20, and reads the data from the field of the same name. The merge task 16 then generates a bit map for the data in accordance with the graphics state and font cache, and again merges the data bit map into the template 28.

5 The merge task 16 continues the steps of identifying variable data areas for the template, retrieving graphics states and character bit maps corresponding to the variable areas, accessing variable data from the merge file 20, and generating bit maps for the variable data, until bit maps have been generated and merged for all of the variable data to be included on the page. When a bit map has been generated for each variable data area, and merged with the template 28,
10 the pagemap is output for printing as shown at 29.

 The merge task 16 then proceeds with printing a second page using the same template and graphics states, but a different variable data record in the merge file 20. To print the second page, the merge task 16 retrieves a "clean" template from the database. Next, the merge task 16 again identifies the name of the first variable data area for that template and retrieves the graphics
15 state of the same name. Then, the merge task 16 reads the data for that field from the second record of the merge file 20, and generates a bit map of the data using the retrieved graphics state attributes and character bit maps. Once the bit map is generated, the merge task 16 merges the bit map into the template by writing the bit map over the template at the location defined by the graphics state.

 The merge task 16 then continues processing in this manner until bit maps have been
20 generated and merged into the template for all of the graphics states reserved for the page. After all of the bit maps for the second page have been merged into the template, the page is printed. The merge task 16 continues, repeating these steps for each record of data in the merge file 20, until all of the variable data records have been printed on a page.

 Figure 2 shows a variable data page printed in accordance with the method of this
25 invention. On this page, the data fields 30 and 32 are static fields which are part of the page template. The data field 34 containing the name "William" is a variable data field. Different names such as Mark or Sam, from the merge file 20, are printed in this field on subsequent pages. The font,

Docket No. 490334.002.C2

angle and color contrast in which "William" is displayed are all aspects of the graphics state which were defined and stored during the steps of the present invention. Data field 36 which contains the number "00467727" is a second variable data area on the page. Again, the data displayed in this area varies on each page, depending upon the contents of the merge file 20.

5 While the method described constitutes a preferred embodiment of the invention, it is to be understood that the present invention is not limited to this precise form, and that variations may be made without departing from the scope of the invention.

What is claimed is:

[illegible]

Docket No. 490334.002.C2

1. A computer implemented method for generating a plurality of bit maps suitable for high-speed printing comprising the steps of:

(a) providing a page description code specification, the page description code specification defining at least one data area, and the page description code further defining a graphics state corresponding to the data area, the graphics state including at least one attribute which controls the appearance of data in the data area;

(b) interpreting the page description code specification, and during the interpretation step, identifying the data area defined by the page description code specification;

(c) upon the identification of the variable data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps;

(d) storing the plurality of character bit maps;

(e) retrieving a variable data item from a plurality of variable data items;

(f) associating the variable data item with the plurality of character bit maps;

(g) generating a variable data bit map for the variable data using the character bit maps; and

(h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items, whereby the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.

2. The computer implemented method of claim 1, wherein the page description code specification represents a template and includes a static data area, and the computer implemented method further comprises the steps of:

executing portions of the page description code specification corresponding to the static data area to generate a template bit map; and

merging each of the plurality of the variable data bit maps into clean copies of the template bit map to create a plurality of merged bit maps.

3. The computer implemented method of claim 1, wherein the identifying step includes the step of detecting predefined characters within a text string defined in the page description code specification.

4. A computer implemented method for generating a reusable template bit map suitable for high-speed variable printing, comprising the steps of:

generating a page description code specification, the page description code specification defining at least one variable data area and at least one static data area;

5 interpreting the page description code specification, and during the interpretation step,

generating a bitmap of the static data area and adding the bitmap of the static data area to a template bitmap;

identifying the variable data area, and

responsive to the identification of the variable data, not adding a bitmap of the

10 variable data area to the template bitmap; and

saving the template bitmap, whereby copies of the template bitmap can be continuously accessed to create a plurality of variable data bitmaps.

5. A computer implemented method of associating a data area defined in a page description language specification with a plurality of variable data items in a merge file, comprising the steps of:

providing a field name in the merge file associated with the plurality of variable data items;

monitoring data areas defined in the page description language specification for a character string matching the field name;

responsive to a detection of the character string matching the field name, associating a data area containing the character string with the plurality of variable data items in merge file.

Docket No. 490334.002.C2

6. The computer implemented method of claim 5, wherein the monitoring step is performed during a step of interpreting the page description language specification.

7. The computer implemented method of claim 6, wherein the step of interpreting the page description language specification includes the step of generating bit maps for data areas defined in the page description language specification.

8. The computer implemented method of claim 7, wherein, responsive to a detection of the character string matching the field name, the step of interpreting the page description language specification includes the step of not generating a bit map for the data containing the character string.

9. The computer implemented method of claim 8, wherein, responsive to a detection of the character string matching the field name, the step of interpreting the page description language specification further includes the step of caching a graphic state for the data area containing the character string, wherein the cached graphic state can be used repeatedly to generate a plurality of variable data bit maps for the plurality of variable data items associated with the field name.

10. A computer implemented method of associating a data area defined in a page description language specification with a plurality of variable data items in a merge file, comprising the steps of:

providing a field name in the merge file associated with the plurality of variable data items;

monitoring data areas defined in the page description language specification for a special character, wherein the special character has a predetermined relationship with a character string matching the field name;

responsive to a detection of the special character, identifying the character string; and

Docket No. 490334.002.C2

- 10 associating a data area containing the special character with the plurality of variable data items in merge file.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231

Abstract

A computer implemented method for generating a plurality of bit maps suitable for high-speed printing includes the steps of: (a) providing a page description code specification, where the page description code specification defines at least one data area, and the page description code further defines a graphics state corresponding to the data area, where the graphics state including at least one attribute which controls the appearance of data in the data area; (b) interpreting the page description code specification, and during the interpretation step, identifying the data area defined by the page description code specification; (c) upon the identification of the variable data area in step (b), applying the graphics state corresponding to the data area to a set of alphanumeric characters so as to generate a plurality of character bit maps; (d) storing the plurality of character bit maps; (e) retrieving a variable data item from a plurality of variable data items; (f) associating the variable data item with the plurality of character bit maps; (g) generating a variable data bit map for the variable data using the character bit maps; and (h) repeating steps (e) through (g) for remaining variable data items in the plurality of variable data items. Thus, the stored character bit maps are used repeatedly to generate a plurality of variable data bit maps.

62414.1

4/24/99

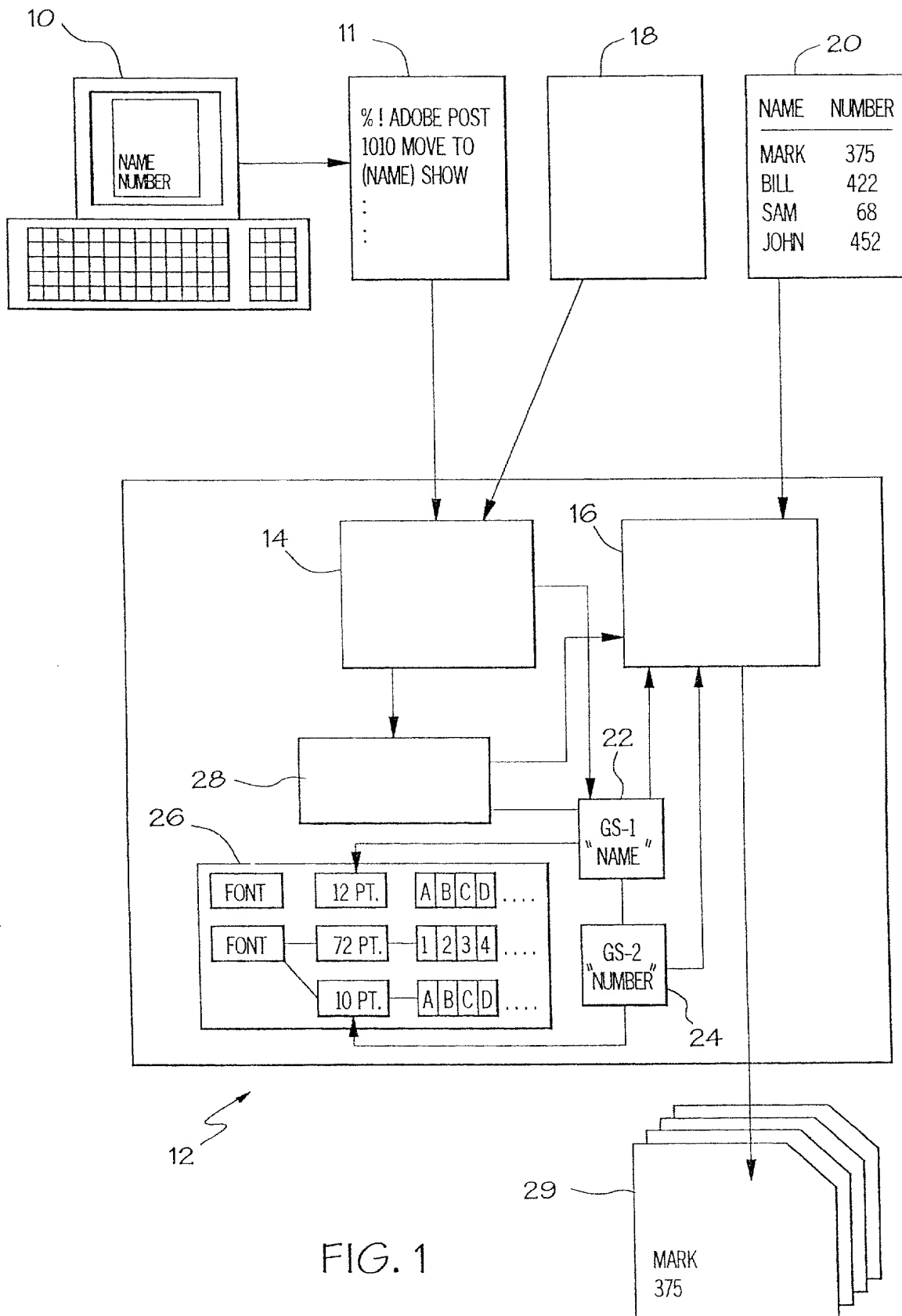


FIG. 1

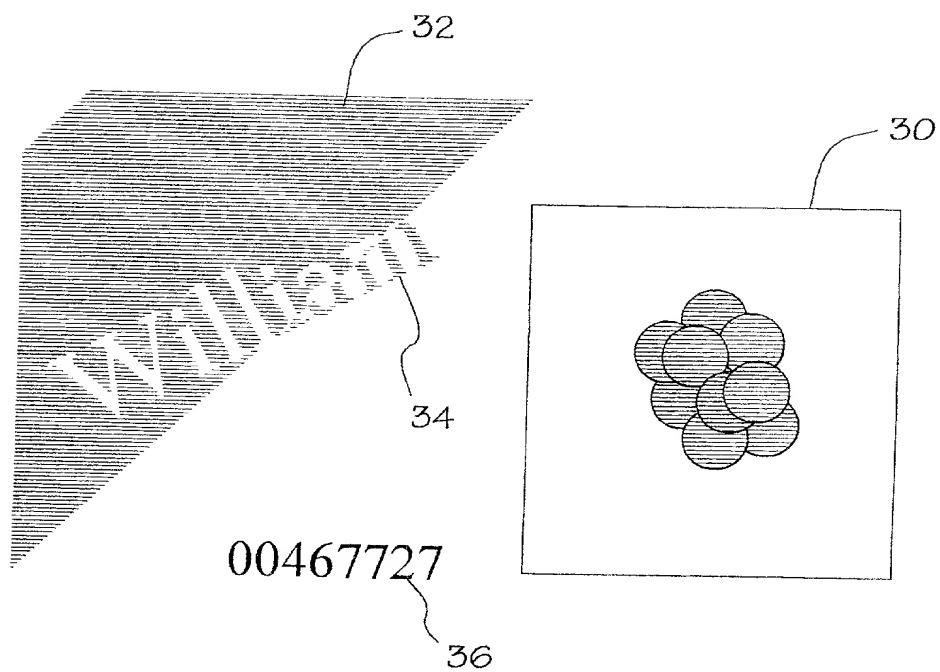


FIG. 2